Design Review 1
ECE 3663
Spring 2013

Billl Devine
Frank Hiemstra
Ryan Hughes
Yatzek Krzepicki

Overview and Introduction

Our approach to implementing the requirements of our design specifications will be detailed in the following design review. This preliminary review covers the first four functions of desired DSP block (figure 1 in the Design Project Assignment pdf) and outlines our approach to the requirements further specified in design review 1, as well as the results of simulations we offer as evidence of functionality.

We will discuss our approach with respect to testing and design for the Sub-Functions under the purview of this design review, as well as proffer our work breakdown schedule for the completion, testing, and documentation of all design and project requirements.

Figure 1.0: ALU Implementation Data Path Block Diagram

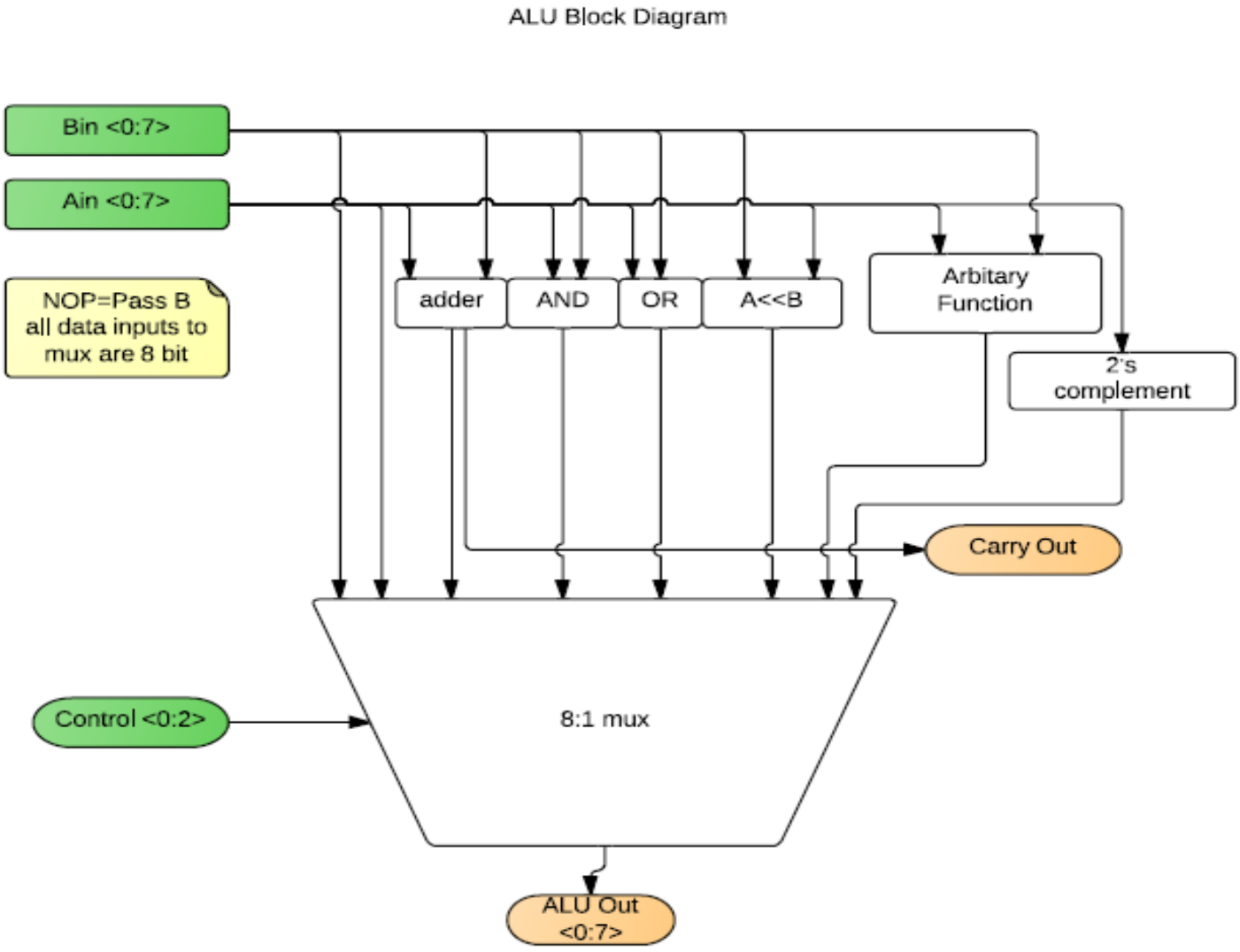


Figure 1.0: the block diagram presented details the data-path of the internal sub-functions of the ALU (PASS A, NOP, ADD, AND, OR, A<<, $A_{rbitrary}$ (A,B), and 2'S COMP A), with functions NOP and PASS A represented as a simple connection of the inputs to the MUX for sake of simplicity, as NOP is equal to PASS B for our specifications.‡

. **The ALU has inputs (shaded green):**

- A_{in} and B_{in}‡ (both clocked & 8 bits)
- Control (unspecified but assumed asychnchonous, 3 bits)

and outputs (shaded orange):

- ALU Out‡ (clocked, 8 bits)
- Carry Out bit from the adder (1 bit, undetermined synchronicity)

All internal data paths have a bit width of 8, and are the respective outputs of the 8 required implemented sub-functions, given inputs A_{in} and $B_{in.}$ We omit the diagram of the data connection for functions not requiring the respective input to be realized.

Progress, Remaining Tasks, and Work Breakdown Schedule

Our team still needs to design and test the remaining ALU functions:

- Shift
- 2Comp
- Add
- Arbitrary

Ultimately we must ensure that all components are working on an individual basis via an iterative testing/simulation process, which our team is still working toward defining and fulfilling. We will want to make sure that the select mechanism works properly as well as that all input/output signal transmission works sequentially and without error. We need to ensure that all registers are working properly and that we receive accurate and reliable output signals when testing our ALU under varying input conditions.

With just over three full weeks remaining prior to our final presentation, we would like as a team for all component-level design work to be completely no later than two weeks from the date of this design review. We will want to allow ourselves at least one full week to integrate our ALU design and continue testing and maximizing efficiency. Ideally, we tackle the remaining components individually and have them done in a relatively short period of time, allowing us an even greater amount of time to test and iterate over our design as a group.

We are planning on using a bidirectional Logarithmic shifter design in order to accomplish the SHIFT function. This will likely be one of the most delay heavy parts of the project, and will have a relatively large area, so we will be spending a large amount of time on this block. For the ADD block, we will be using a mirror adder design, which saves power and decreases gate area, therefore decreasing delay. This will

‡ In the final project design specifications, the 8-bit data-steam ALU Out is recycled and piped to Bin, thus PASS B is equivalent to no change on the output.

also be a delay heavy block, so the clock speed will likely depend on our ability to decrease delay in these two functions.

2COMP will be designed as a set of 8 parallel inverters, fed into an adder which adds 1 to the inverted input. The arbitrary component of our design is something our group is still considering. We have narrowed it down to a multiplier or a divider, but we have not decided for sure which one we will include.

By END of Week 4/1/13:

- Finish SHIFT (Bill Devine, Frank Hiemstra),
- Finish ADD (Ryan Hughes, Yatzek Krzepicki)
- Specify design variables (team)
  - Define $A_{rbitrary}(A_{in}, B_{in})$
  - Clock status of bits with undetermined synchronicity

By START of Week 4/8/13:

- Start on 2COMP (at least have schematic done, Ryan Hughes, Bill Devine)
- Start on arbitrary function (Yatzek Krzepicki, Frank Hiemstra).
- Begin integration of completed components in preparation for large scale testing (Team)

By END of Week 4/8/13:

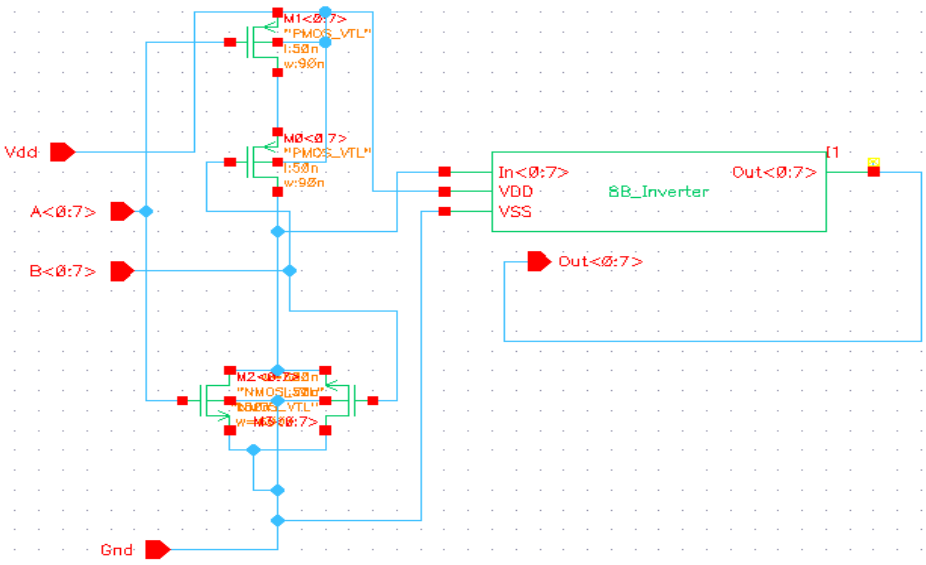
- Done with all Schematics and individual component testing (Team)
- Begin integration and large scale testing/simulation (Team)

Appendix A:
Schematics of ALU Sub-Functions and Components

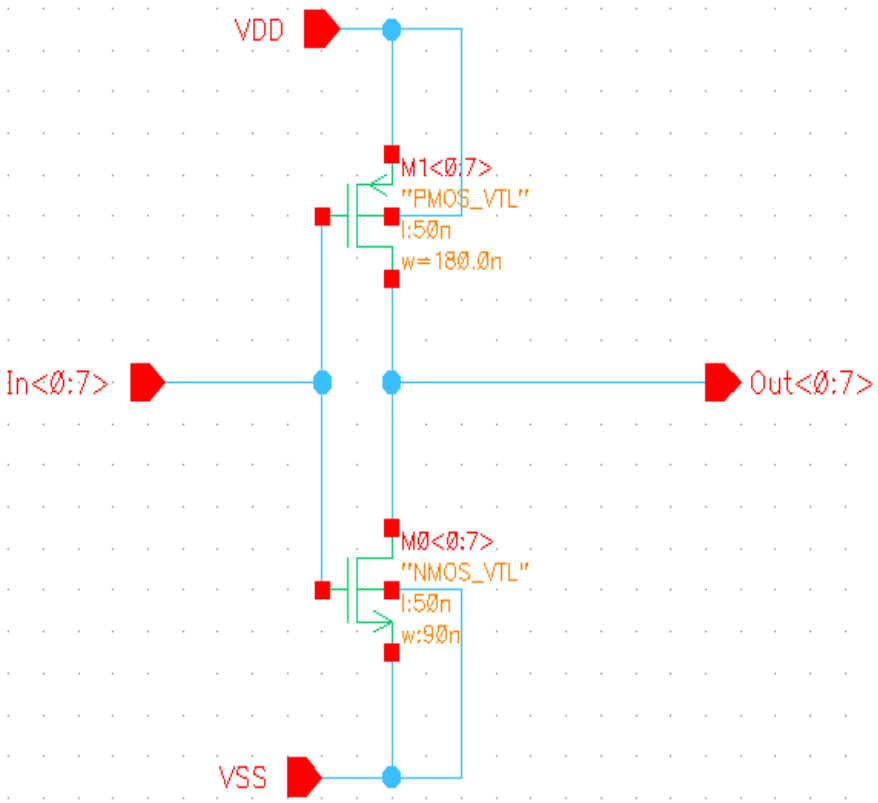
Appendix A.1.1: implementation schematics – Or

There are two 8-bit input pins (marked A<0:7> and B<0:7>) and one 8-bit output pin (marked Out<0:7>). There are also Vdd and Ground connections. The pull-up network consists of two 8-bit PMOS transistors in series, each having width of 90 nm. The pull-down network consists of two 8-bit NMOS transistors in parallel, each with width of 600 nm. The inverter block is expanded in **Sch. 1.2.b**.

Schematic 1.1.a: ALU Sub-Function OR (8-Bit Hierarchy)



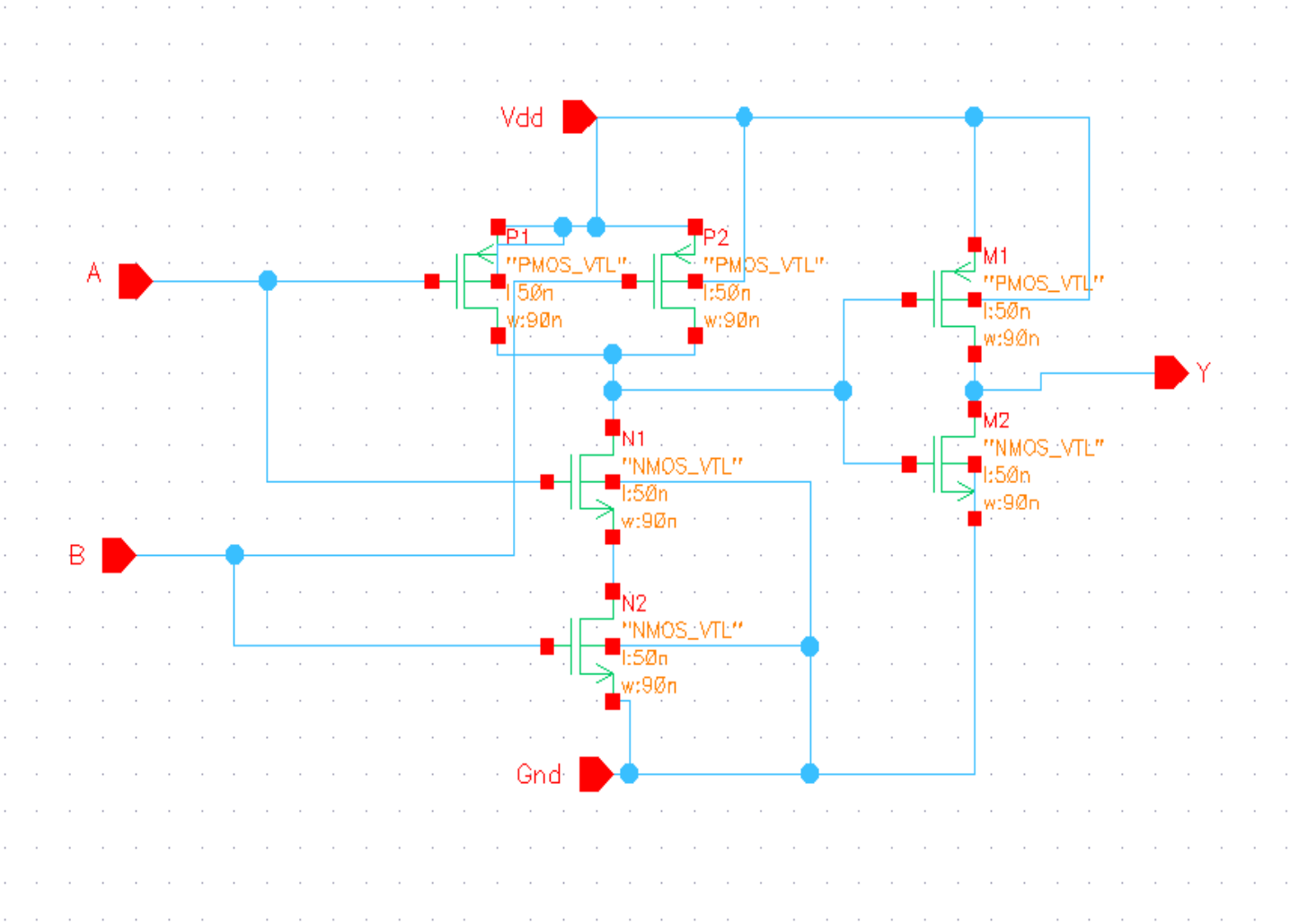
Schematic 1.2.b: Inverter Block Structure (parallel 8-Bit)



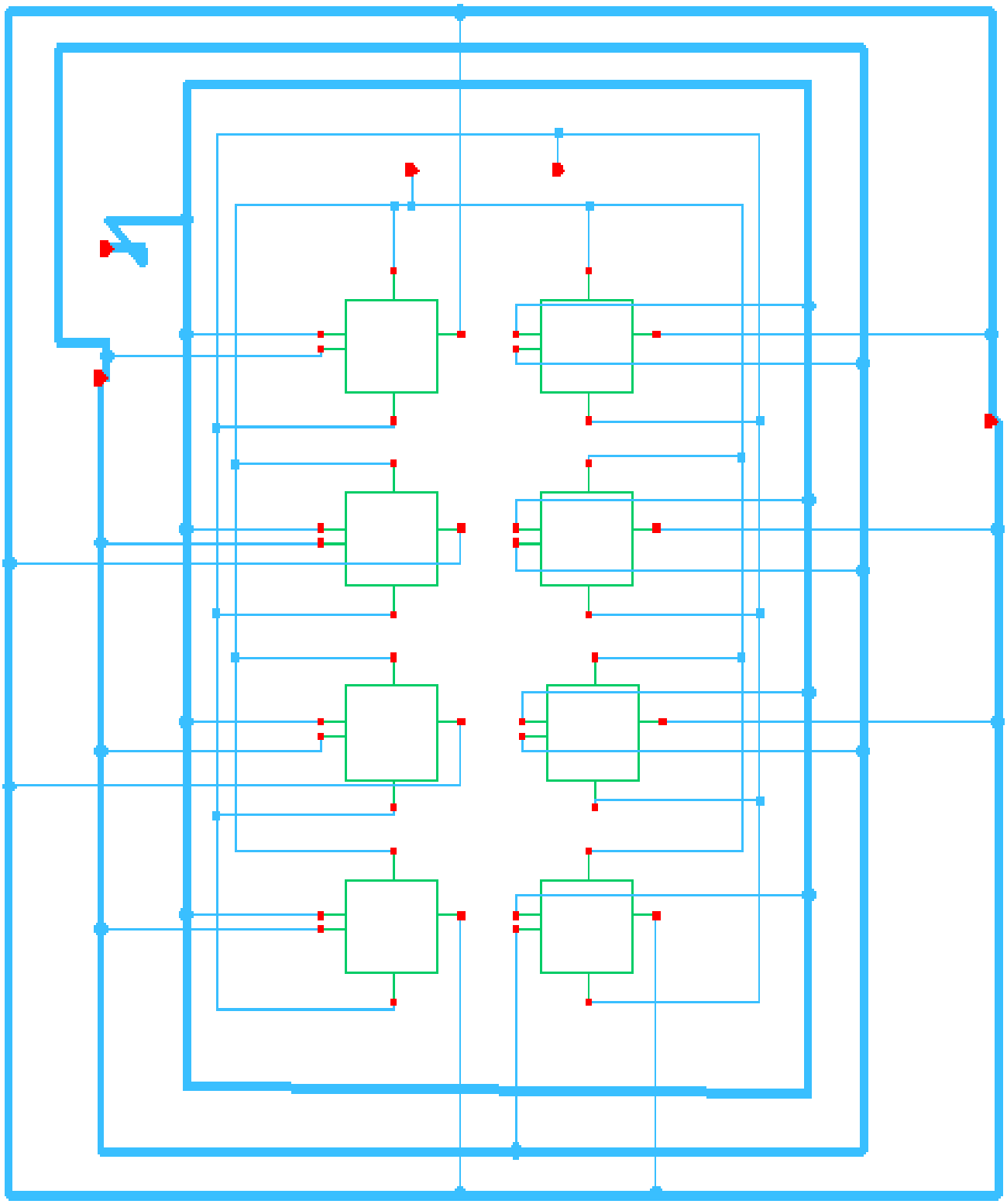
Appendix A.1.2: implementation schematics - And

AND by a CMOS NAND gate with an inverter on the output. Simple functions, such as AND, OR, and PASS A, will not almost certainly not be on the critical path of the ALU, This is likely to be made up of more complicated functions such as shifting and adding. This design consists of 8 parallelized 1-bit AND gates, which combine to form a 1-byte AND gate. Each individual 1-bit NAND gate has a Pull-Up Network containing 2 parallel PMOS transistors and a Pull-Down Network of 2 series NMOS transistors. Each 1-bit NAND is then connected to a classic CMOS inverter. The schematics for the entire 1-byte gate as well as the 1-bit gate are shown below (**Sch. 12.a&b)**,. In the schematic of the overall layout of the 1-byte gate(**Sch 1.2.a0**, the thicker wires represent an 8-bit bus, while the thinner wires going to each individual block (representing a single bit AND) are breakouts from this bus. In this diagram (**Sch 1.2.b)**, the 4 blocks in the left column represent bits 0-3, while the 4 in the second column represent bits 4-7.

Schematic 1.2.a: ALU Sub-Function AND (Single Bit Implementation)



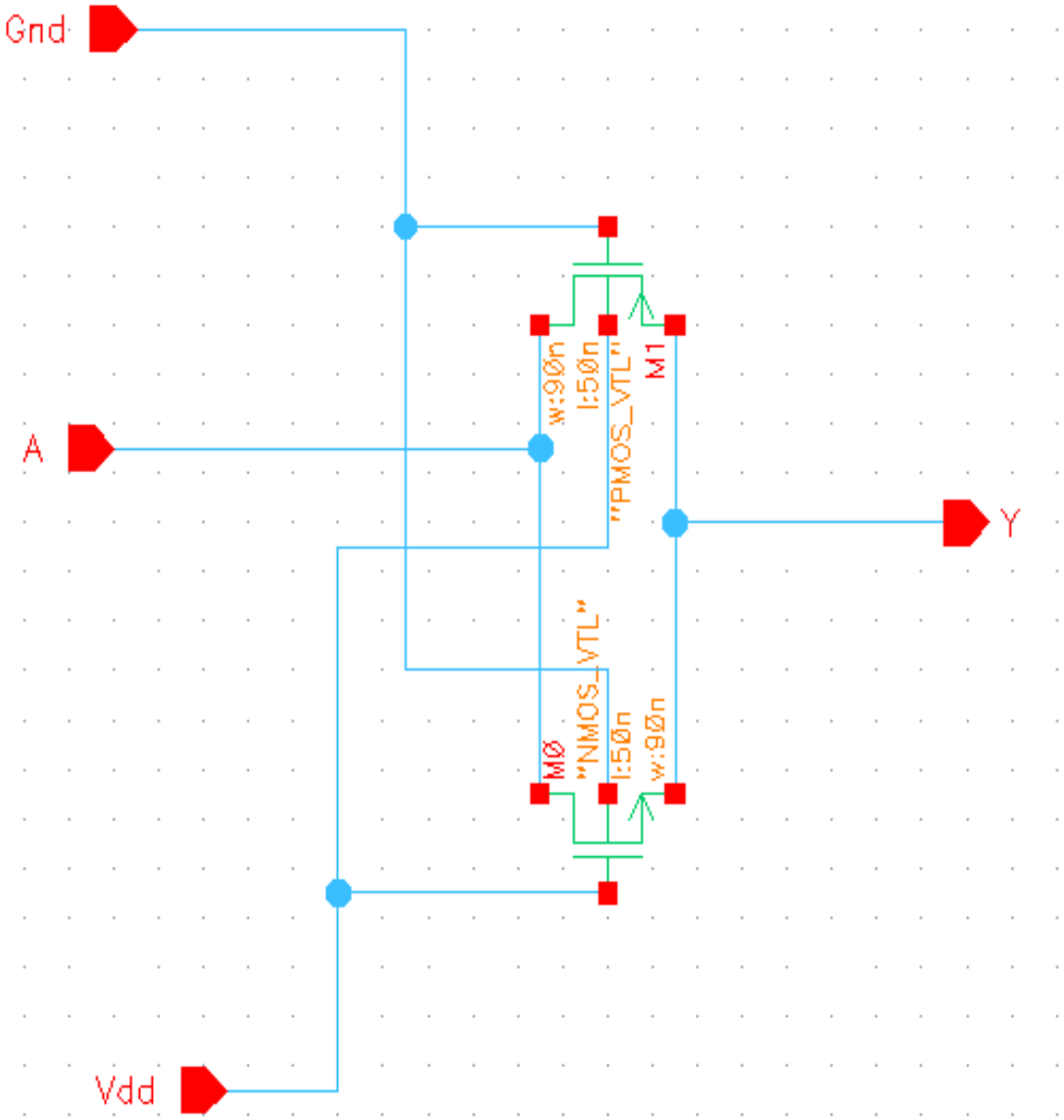
Schematic 1.2.b: ALU Sub-Function AND (8-Bit Hierarchy)



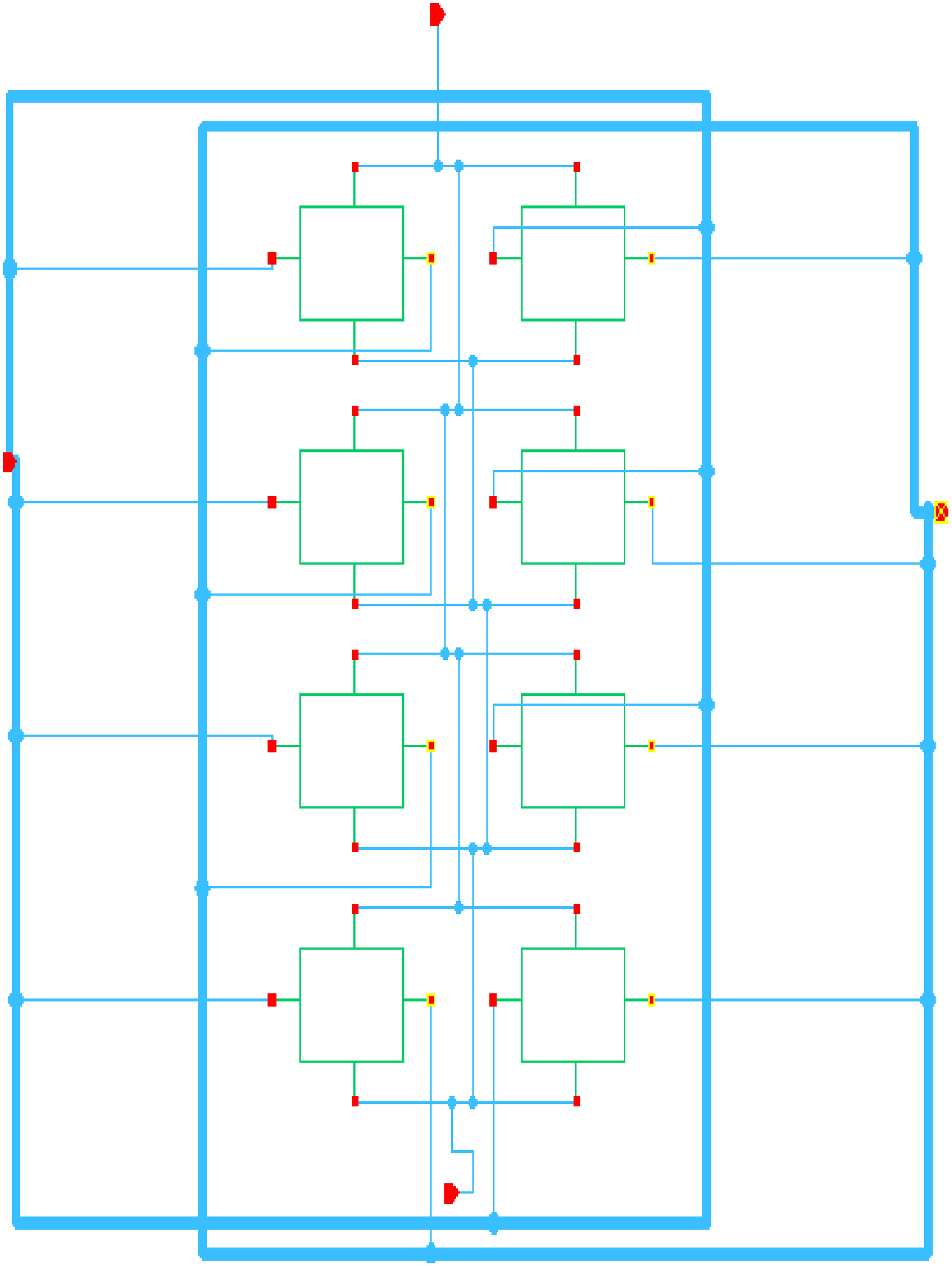
Appendix A.1.3: Implementation Schematics - Pass A

We use transmission gates in order to ensure that the outputs are at full logic level, functionally making Pass A 8 parallel transmission gates. We implement using the CMOS transmission gate, as this is unlikely rate-determining.. Both the single transmission gate and the 8 bit wide transmission gate schematics are shown below. As in the 8-bit AND gate above, the left column of the 8-bit PASS A corresponds to the 4 Least Significant Bits, while the right column represents bits 4-7. Thick wires represent an 8-bit bus, and thin wires coming from a bus represent breakouts (individual bits) from that bus.

Schematic 1.3.a: Single Bit Pass A Transmission Gate



Schematic 1.3.b: ALU Sub-Function Pass A Transmission Gate (8-Bit)

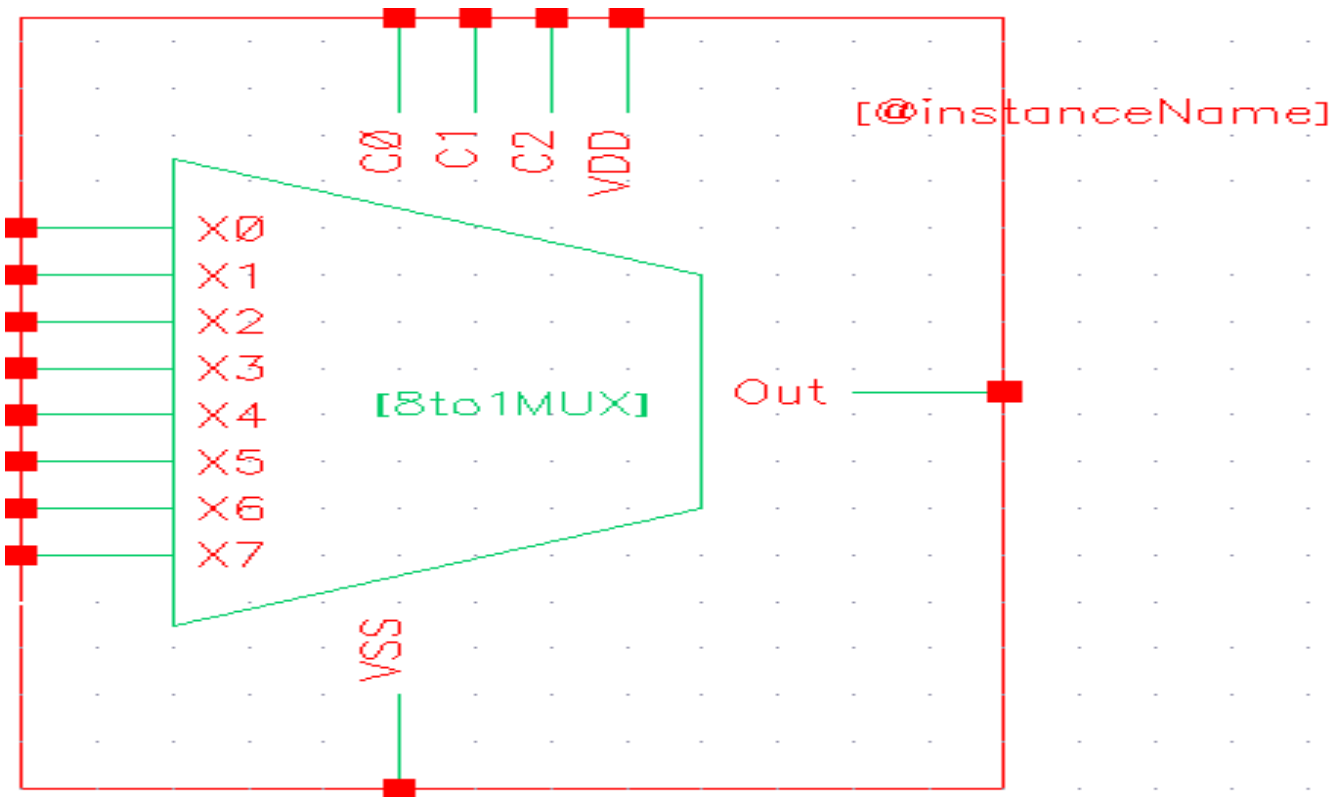


Appendix A.1.4: Implementation Schematics – 8 to 1 Mux

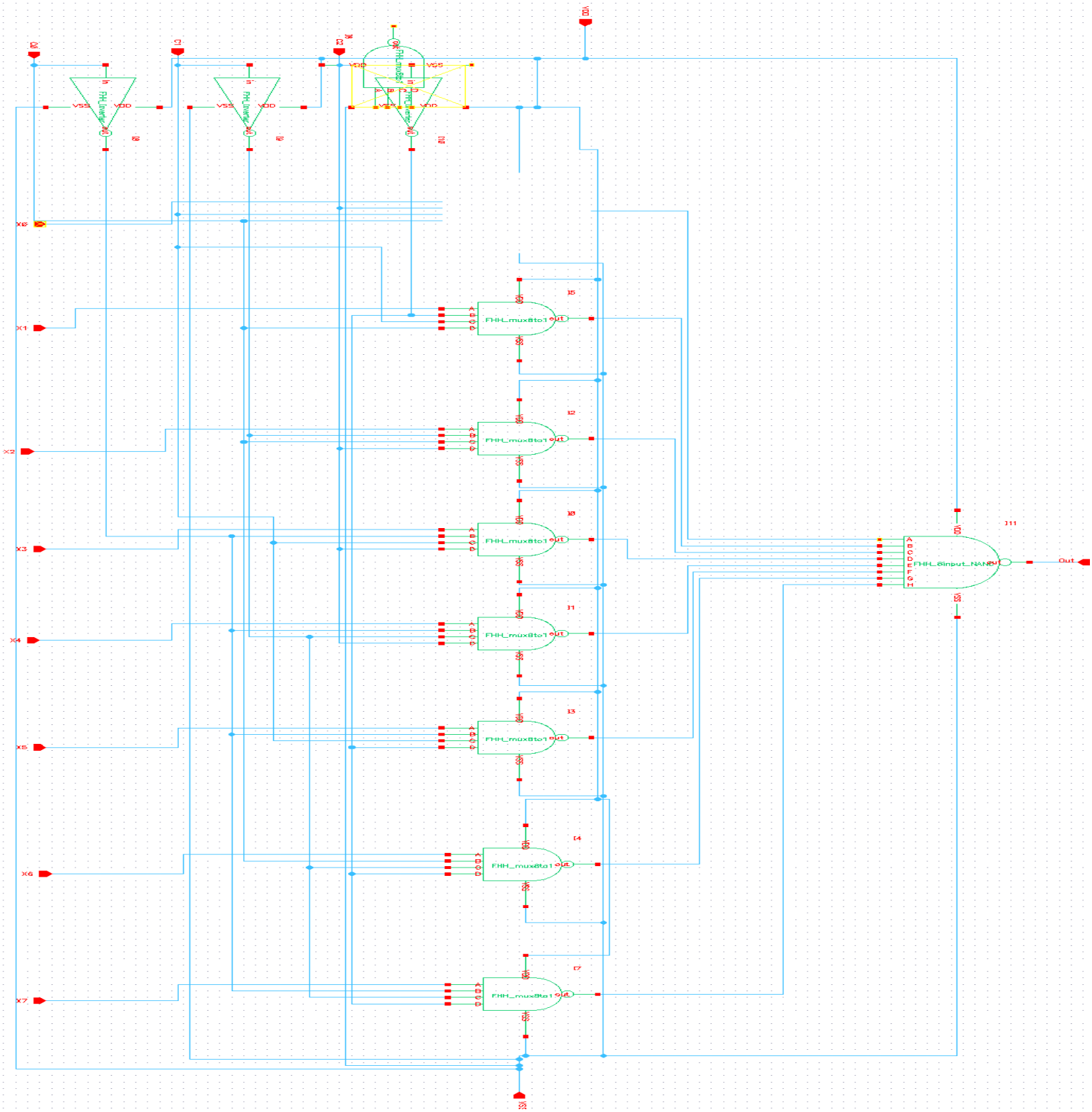
This block is a relatively simple implementation of gate logic to realize a multiplexer; 3 control bits <c0:c2> select from the output of one of 8 functional blocks on <in0:in7> which is passed to ALU out <0:7>. This is implemented using transistors to realize logic gates allowing NAND-level iteration over the control bits c0:c2, each which will ultimately represent a call to one of the block level functions detailed in figure 1.1 and as specified our project requirements.

In plain terms, our multiplexer takes the data output of one of the previously detailed Sub-Functions (**Fig 1.1, Appendix A),** an 8-bit internal data path for each of the functions, and inputs it on pins x0<1:7> through x7<0:7>, which are identified and called by c<0:2>. For simplicity's sake, let the index of the X_n pin correspond to the decimal value of the unsigned binary numeral c2 c1 c0. The data path is subject to a bitwise AND across the inputs (passing parallel instances of the control bits) such that one input path is unaffected and the rest are zeroed out ergo Out becomes a bitwise OR applied across this manipulated input data path, such that the data on the input pin of index selected c<0:2> is passed to Out.

Schematic 1.4.b ALU 8-to-1 Multiplexer Block-Level Diagram (8 bits across in0-in7)



Schematic 1.4.a: ALU 8-to-1 Multiplexer Gate-Level Schematic



Appendix B: Transient Analysis and Testing Strategy

Within Cadence, the Virtuoso Analog Design Editor (ADE L) provides a good method of performing rudimentary analysis and simulations on circuits. This tool allows us to run DC operating point and transient analysis on specific signals within our test bench setup. Each gate will be tested within a test bench schematic for versatility in adjusting testing parameters and expediency as we test multiple devices.

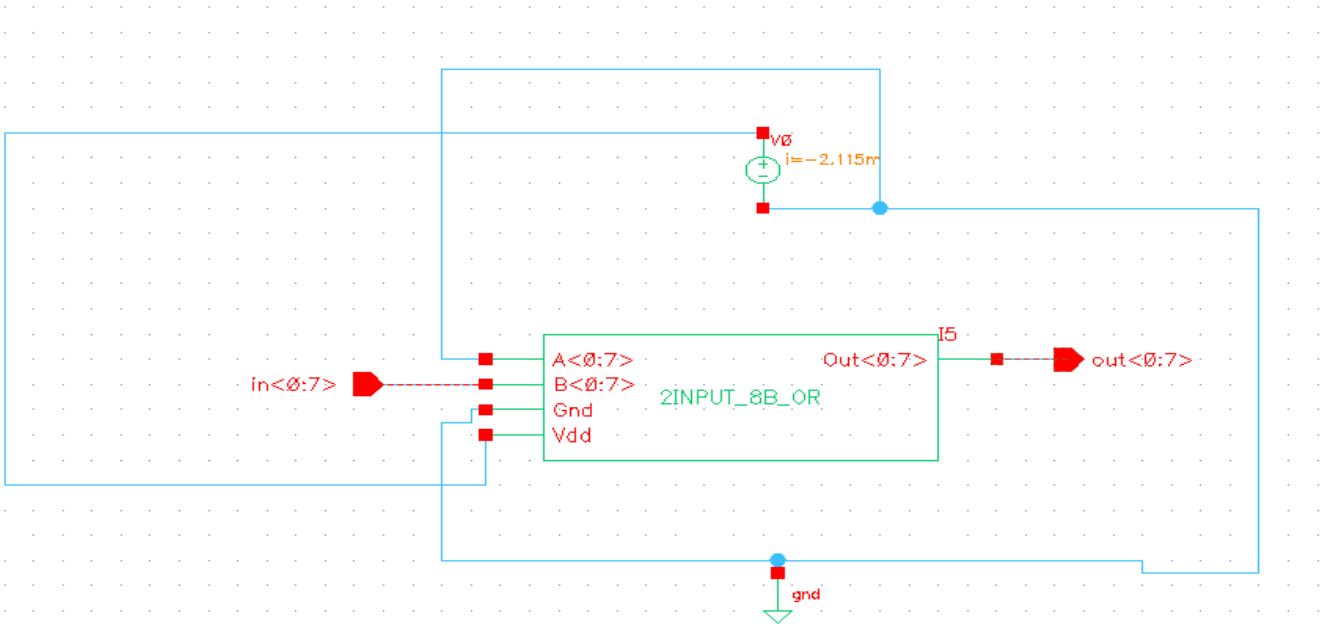
To allow for more expedient and efficient simulations, our team will make use of OCEAN in further simulations. OCEAN will allow us to iterate our simulations with varying design parameters in a much quicker fashion than utilizing the ADE L user interface within Cadence. Our team still needs to research and become more comfortable with OCEAN before using it in simulations, so for this design review we used ADE L for transient simulations.[1]

[1] For this design review, it was sufficient to simulate only 1-bit of a bitwise function (such as AND, OR, and PASS A) as a matter of convenience

Appendix B.1.1: Or Sub-Function Transient Analysis

Fig B.1. below shows the test bench utilized in testing the OR gate. In the test bench schematic, we provide ground and Vdd nodes to connect to the 8-bit, 2-input OR gate block, and we also define 8-bit input and output pins. On the input pin we are able to define a DC pulsing stimuli to test functionality of the gate with the other gate input held either high or low. A transient Analysis is run on these settings.

Figure B.1: Test-Bench Configuration for the Or Gate.

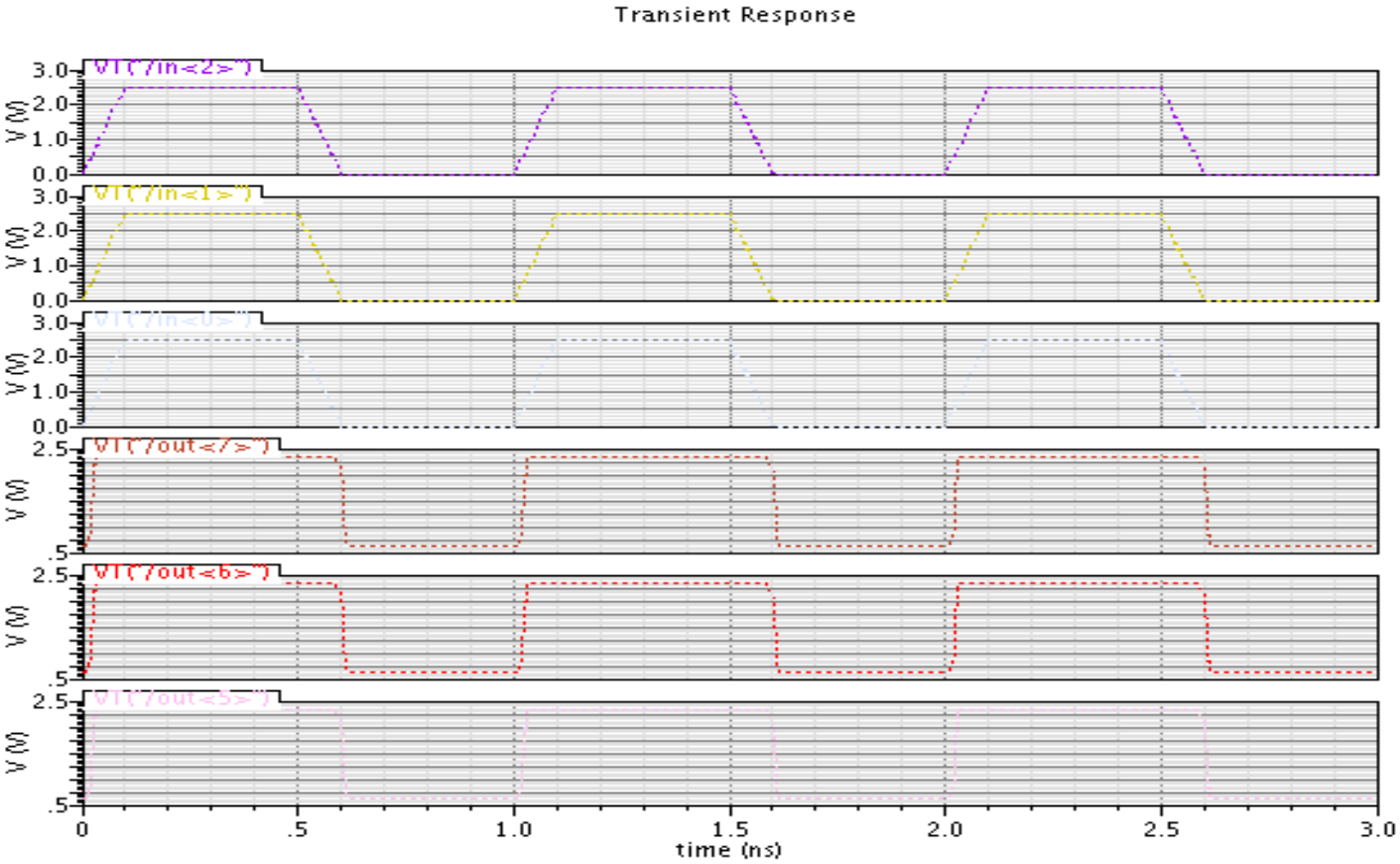


In testing the functionality of the OR gate, we began with the A<0:7> input of the gate tied to ground, which was an element of the testing process in which one input was held low while the other (B<0:7>) fluctuated from low to high via a DC pulse stimuli across the B<0:7> input. The resultant DC transient analysis plot shows that condition 00 (A<0:7> is low and B<0:7> is low due to the state of the DC pulsing stimuli) results in an output of 0, as expected, and condition 01 (A:<0:7> low and B<0:7> high) results in an output of 1 (an output voltage of around 2.35 VDC), as expected.

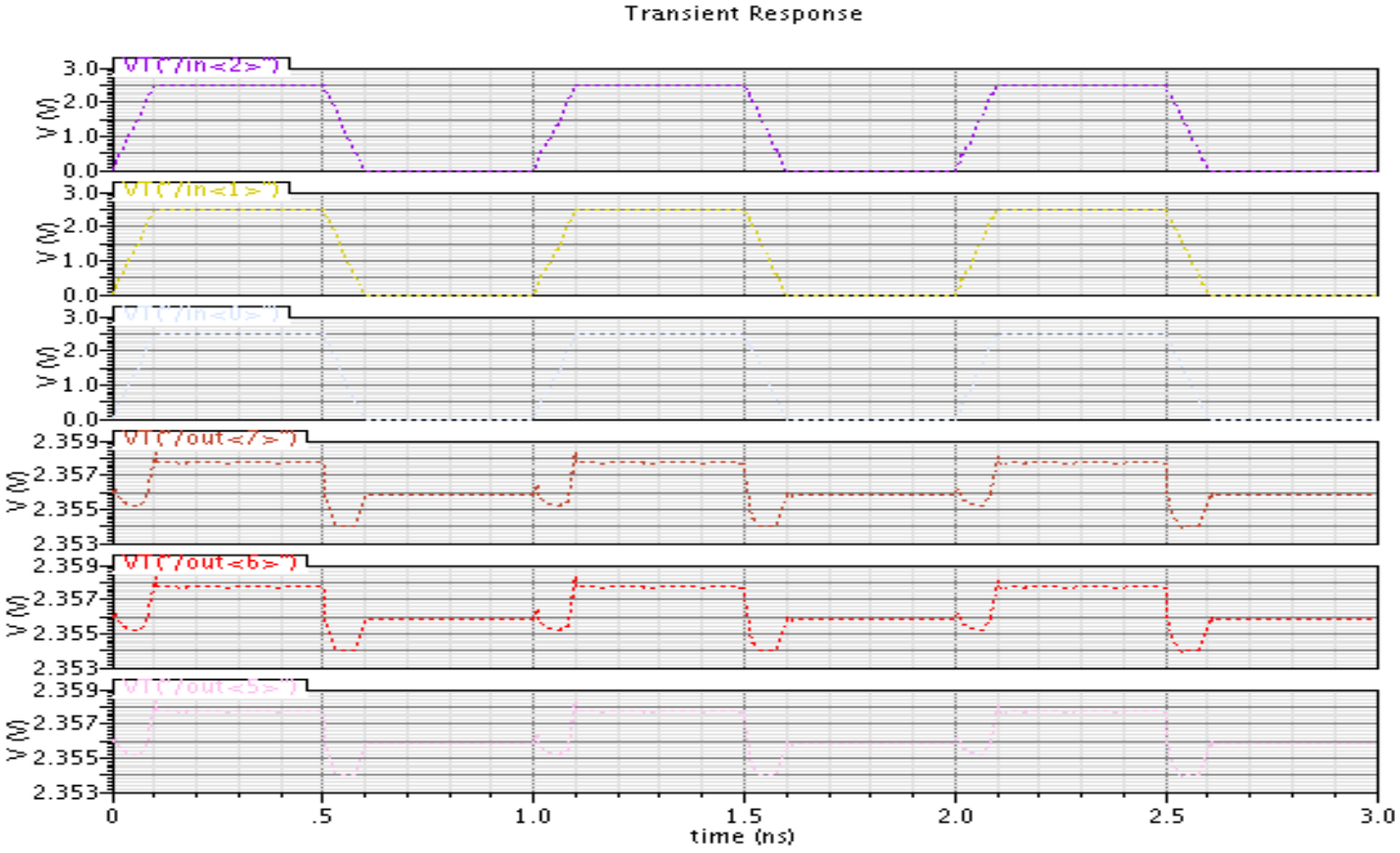
The OR gate was also tested such that A<0:7> was tied to Vdd, providing 2.5 VDC across all eight input bits. While holding the A<0:7> input high, the same stimuli was added across the B<0:7> input. What resulted was an output of 1 regardless of the state of the DC pulsing stimuli, as expected (conditions 10 and 11 would both result in an output of 1 with an OR gate). Testing in this manner however resulted in an output that was still reliable (always remained high within some threshold [approximately 2.35 VDC – which will be fine as long as the rest of our design takes this small variance into account]) but lacked the stability provided in the first testing method (i.e., with each transition of the DC pulse stimuli, there is a very minor voltage droop). Our team is exploring methods to improve the output signal quality throughout each transition in this case; however, we are confident that it will not be problematic in our overall design.

Both plots shown below demonstrate the functional behavior of the OR gate. Each plot below compares three input bits and three output bits based on the testing scenarios described above. The first plot involves A<0:7> tied to ground and the output responding to the stimuli on the B<0:7> input vector. The second plot involves B<0:7> tied to ground and the output responding to the same stimuli on B<0:7>.

Graph B.1.1: Graph of OR Block Transient Analysis Simulation Results, A tied to GND



Graph B.1.2: Graph of OR Block Transient Analysis Simulation Results, A tied to VDD

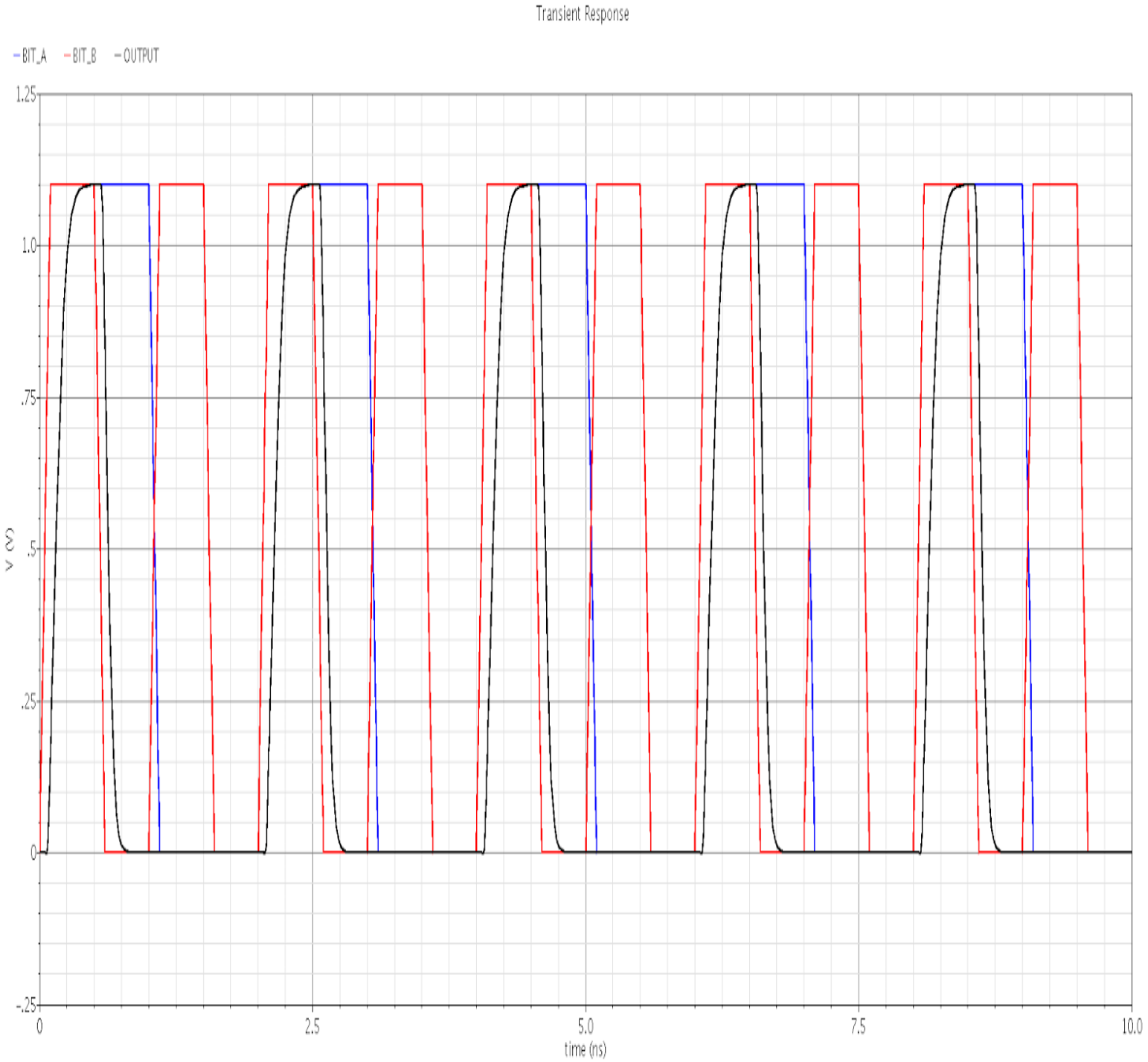


Appendix B.1.2: And Sub-Function Transient Analysis

A Test Schematic with a voltage source (for the power supply) and some input stimuli was set up to test the circuit. The input stimuli were set to be pulse inputs, but since we want to test all 4 input scenarios (00, 01, 10, 11) as efficiently as possible, we simply set the period of one of the input pulses to 2nS and the other to 1nS.

Therefore, one input pin will be held high while the other does a full cycle, then this will repeat with the first pin held low. This is well illustrated in **Graph B.2**, which shows that the gate does work, with an approximate delay at this point in the design process of 0.1 nS. We have not yet optimized the transistor widths for this gate.

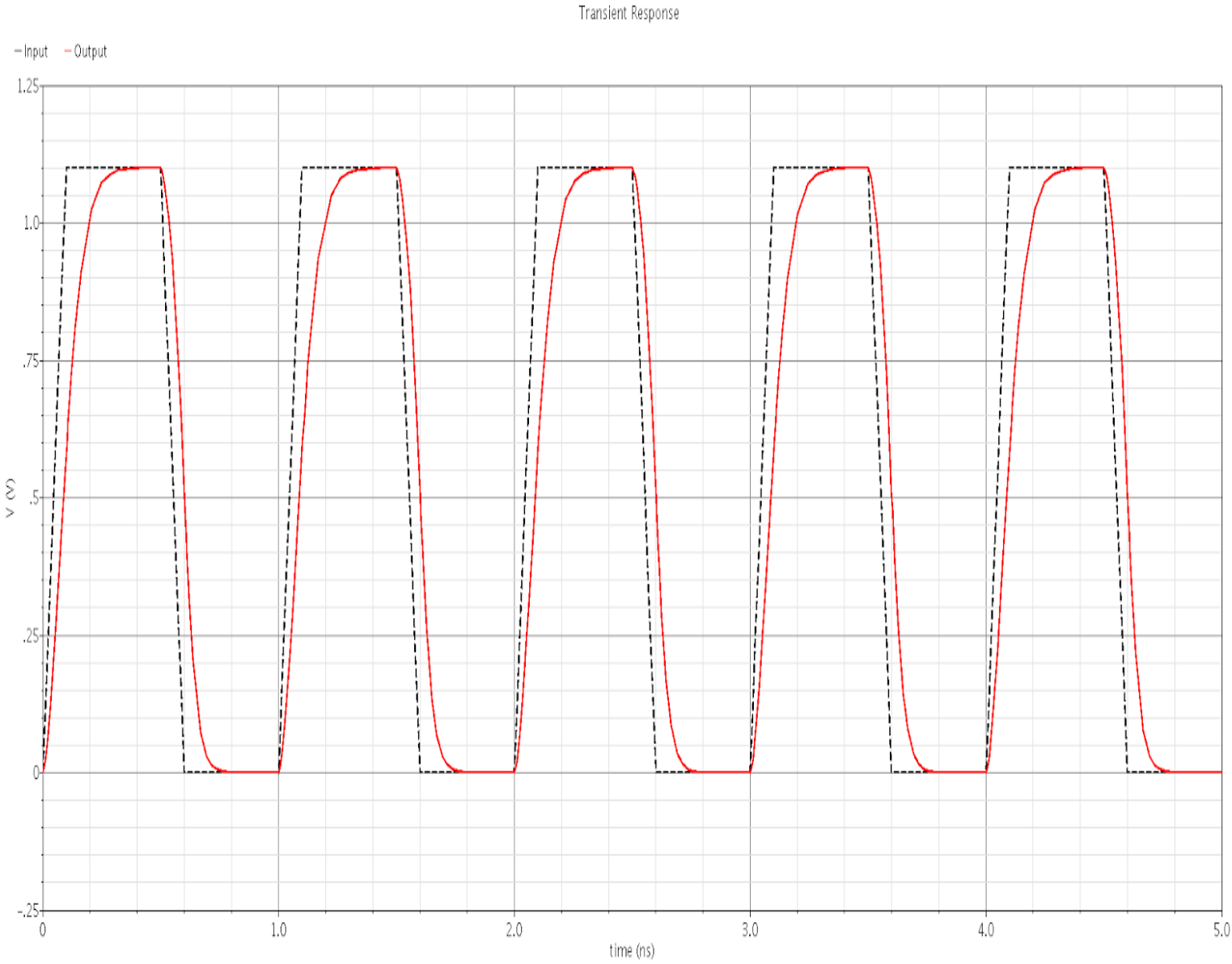
Graph B.2: Graph of AND Block Transient Analysis Simulation Results



Appendix B.1.3: Pass A Sub-Function Transient Analysis

In order to test the PASS A gate, we only had to verify that one bit worked in order to verify that the larger, 8-bit gate would work. As the 8 bit gate is constructed from 8 single bit transmission gates in parallel, it is reasonable to suspect that the entire 8 bit system will work if the single bit gate works. Like the AND gate, we set up a transient simulation with a 1.1 V power supply and a pulse on the input pin. This pulse was set up to have a period of 1 nS. The output waveform should follow the input waveform. This is verified on the graph below. The system currently has a delay time of approximately 0.1 nS. As we have not yet optimized the transistor characteristics, we expect to be able to reduce this delay.

Graph B.3: Pass A Transient Simulation Results

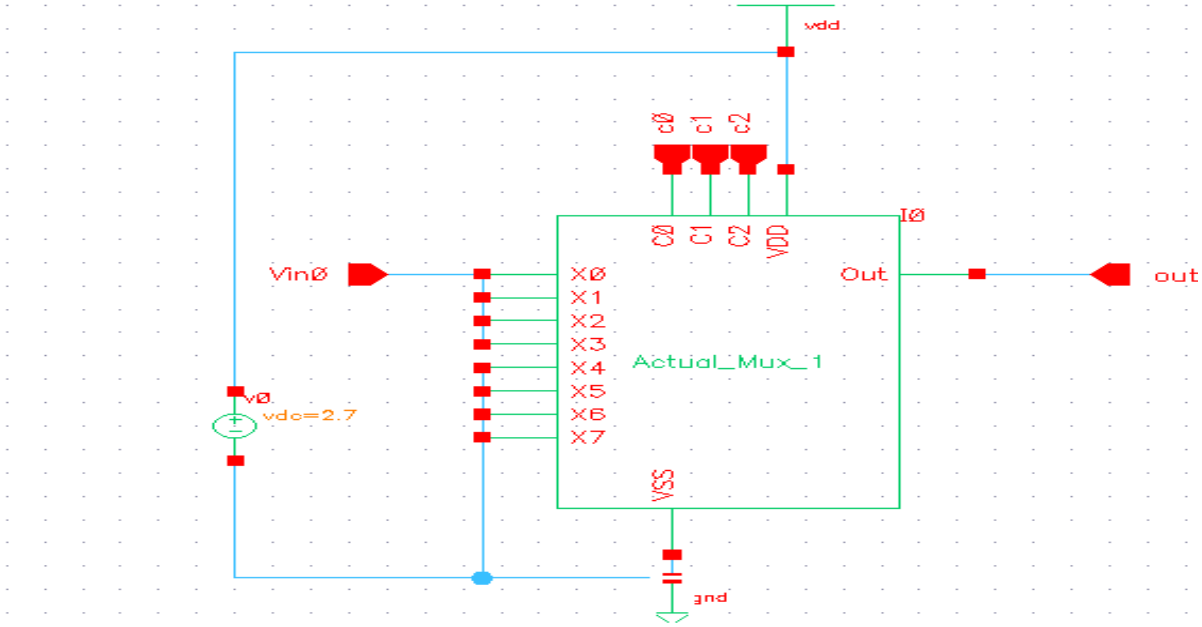


Appendix B.1.4: 8-to-1 Multiplexer Transient Analysis

To test the multiplexer, we will use one bit (expressed as both 0 and 1) on the input in lieu of the full 8-bit data stream, as it proves sufficient to prove functionality.

To simulated and test the MUX, we iterate over control bits on 0-7 (as a 3 digit binary number) using periodic pulsing sources such that the control should cycle over all inputs. For each iteration, changes on the input should only be manifest when the index is called by the control bits, otherwise no signal should manifest on the output, as all other bits are grounded for testing purposes. The overall test-bench layout for the multiplexer is shown below in **Graph B.4** below.

Graph B.4.1: Test Bench Layout for the Multiplexer



Graph B.4.2: 8 to 1 Mux Transient Simulation Reslts